**GIT**

**Version Control System**

Version control is a system that records changes to a file or set of files over time so that you can recall specific versions later.

**Types of Version Control Syste**

1. Local Version Control System.
2. Centralized Version Control System.
3. Distributed Version Control System.

**WHAT IS GIT**

* Git is a DevOps tool used for source code management.
* GIT is a free and opensource distributed version control system.
* Used to handle small to very large project efficiently.
* Git is used to tracking changes in the source code.
* Enable multiple developers to work together.
* Git like tools AWS CodeCommit, Mercurial, Helix Core etc.

**GIT BENEFITS**

* Work on snapshots.
* Every operation is local.
* Git has integrity
* Git generally only adds data.

**SOURCE CODE MANAGEMENT**

1. CVCS = CENTRALIZED VERSION CONTROL SYSTEM (OLDER METHOD)
2. DVCS = DISTRIBUTED VERSION CONTROL SYSTEM (CURRENTLY USING EX. GIT)

|  |  |
| --- | --- |
| **1**  WORKING  AREA | **2**  STAGING  AREA |
| LOCAL  REPO  **USER B** |

**3**

**3**

**USER A**

**PULL**

**PULL**

**PUSH**

**PUSH**

**GitHub**

**CENTRAL REPOSETRY**

|  |  |
| --- | --- |
| **1**  WORKING  AREA | **2**  STAGING  AREA |
| LOCAL  REPO |

IQ – DIFFERENCE B/W CENTRALIZED VERSION CONTROL SYSTEM AND DISTRIBUTED VERSION CONTROL SYSTEM?

ANS - The main difference between centralized and distributed version control is that, in centralized version control, the versions are saved in the remote repository, while in distributed version control, versions can be saved in the remote repository as well as in local repositories of the local.

**3 STAGES OF GIT**

1. **MODIFIED**(If file was changed since it was checked out but has not been staged, it is **modified)**
2. **STAGED** (If it has been modified and was added to the staging area, it is **staged)**
3. **COMMITED**(If a particular version of a file is in the Git directory, it’s considered **committed)**

**GIT INSTALL**

Yum install git -y

Git –version

mkdir /mywork

cd /mywork  
git init ( to initialize git workspace)

vim deepak

git status ( to check code status)

git add deepak ( to add file in staging are)

git commit -m “my first commit” deepak (m= message, msg aap apne hisab se kuch bhi de skte ho)

git log ( to view all commits)

git log –decorate (makes git log display all of the references (e.g., branches, tags, etc) that point to each commit.)

git log –oneline (to display the output as one commit per line**)**

git log –oneline --decorate

**GIT Config. For particular work space** (Ye configuration dusri directory ke lie kaam nhi krega)

Git config user.name “Deepak Sharma”

Git config user.email [deepakkrsharma29@grras.com](mailto:deepakkrsharma29@grras.com)

**GIT Config. For Global Setting** (Ye configuration globally kaam kregi)

Git config --global user.name “Deepak Sharma”

Git config --global user.email [deepakkrsharma29@grras.com](mailto:deepakkrsharma29@grras.com)

**GIT Config. For system Setting**

Git config --system user.name “Deepak Sharma”

Git config --system user.email [deepakkrsharma29@grras.com](mailto:deepakkrsharma29@grras.com)

**Working on GITHUB**

Git remote add origin LINK (LINK = Link of github repository, ogigin name ke variable me hum link ki value store krvare h taki bar-bar link past nhi krna pde )

git clone centralized\_repo\_link (is command se GitHub (central repo.) ki all files local machine pe clone/copy ho jati h)

git push origin master (origin = aap origin ki jagha kuch bhi name de skte ho, master = master branch)

git pull origin (github ka data local machine pe lane ke lie )

NOTE:- in this command it will ask username and password then you can give username and password (but nowdays it don’t accept passwords that’s why you have to generate token) otherwise you can go to GitHub website and ->profile->setting->developer setting->personal access tokens->generate new token And then generate token by selecting some checkboxes according your requirement.

Then copy that token number and past that token when it ask for password after giving Username.

**BRANCHES**

Branches **allow you to develop features, fix bugs, or safely experiment with new ideas in a contained area of your repository**. You always create a branch from an existing branch. Typically, you might create a new branch from the default branch of your repository.

NOTE:- Branches me 1 Master branch hoti h baki Feature branch hoti h.

Master Branch

Feature1 Branch

Feature n Branch

**BRANCH COMMANDS**

git branch (to list all branch)

git branch kgf2 (to create a new feature branch)

git branch -D kgf2 (to delete a branch)

git checkout BRANCH\_NAME (to switch one branch to another)

**MERGE**

git merge kgf2 (feature branch ka data master branch me add krne ke lie kiya jata h)

NOTE:- merge hamesa master branch se kiya jata h. Feature branch ka data master branch me add krne ke lie merge use kiya jata h.

**REBASE**

git rebase master (feature branch me master branch ka data update krne ke lie kiya jata h)

NOTE:- rebase hamesa feature branch se kiya jata h. Agar master branch ne feature branch banane ke bad master branch me kuch update kya h to use feature branch me update krne ke liye feature branch ko rebase karna hoga.

**IQ – DIFFERENCE BETWEEN MERGE AND REBASE?**

**ANS –**

**MARGE =** merge hamesa master branch se kiya jata h. Feature branch ka data master branch me add krne ke lie merge use kiya jata h.

**REBASE =** rebase hamesa feature branch se kiya jata h. Agar master branch ne feature branch banane ke bad master branch me kuch update kya h to use feature branch me update krne ke liye feature branch ko rebase karna hoga.

**PASSWORD CREDENTIAL SAVE PERMANENTALY**

Agar aap bhi chahte h ki aapke system pe bhi bar-2 credentials na puche to aap credentials apne system pe store krva skte h is command ke through. Is command se kya hoga ki jab bhi aap is command ko run krane ke bad username or password denge to use system save kr lega fr dobara nhi puchega.

git config--global credential.helper store

file location cd /root/.git-credentials

if you delete this file then system will ask you username and password again.

**GIT HOOKS**

Git hooks are **scripts that run automatically every time a particular event occurs in a Git repository**. They let you customize Git's internal behavior and trigger customizable actions at key points in the development life cycle.

By default system me kuch hooks hote h

Hooks location cd .git/hooks

(You can get hooks code from google.)

**Yhan hum 1 Post Commit hook configure krte h**

**POST COMMIT:-** The post-commit hook is called immediately after the commit-msg hook. It can’t change the outcome of the git commit operation, so it’s used primarily for notification purposes.

* Search on google “git hook for post commit”
* <https://www.atlassian.com/git/tutorials/git-hooks>
* vim .git/hooks/post-commit (is file ki command abhi nhi chlegi kyonki file ko abhi execute ki permission nhi h)

#!/bin/bash

git push origin master

* chmod a+x post-commit (is command se execute ki permission mil jayegi)
* ab hum master pe kuch bhi commit krenge to automatically github pe push ho jayegi.

**MERGE CONFLICT**

A merge conflict is **an event that takes place when Git is unable to automatically resolve differences in code between two commits**. Git can merge the changes automatically only if the commits are on different lines or branches.

If you face **MERGE CONFLICT** then use this tool

* git mergetool
* git commit -m “final commit”

**ROLE BACK/RESET**

GIT RESET 3 METHODS (bydefault reset method is MIXED)

1. SOFT
2. MIXED
3. HARD
4. SOFT (soft reset me file commit area(locak repo.) se staging area me aa jati h jise dobara commit kiya ja skta h)

* git reset --soft HEAD~
* git log --oneline --decorate

1. MIXED (mixed reset me file commit area or staging area dono jagha se working area me aa jati h jise dobara add or commit kiya ja skta h)

* git reset --mixed HEAD ~
* git log--oneline –decorate

1. HARD (hard reset me file sabhi jagha se delete ho jati h like working area, staging area or commit area(local repo.)

* git reset --hard HEAD ~
* git log --oneline --decorate

**SQUASH**

To "squash" in Git means **to combine multiple commits into one**. You can do this at any point in time (by using Git's "Interactive Rebase" feature), though it is most often done when merging branches. Please note that there is no such thing as a stand-alone git squash command.

You can merge several commits into a single commit.

Git squash krne se feature branch ke jitne bhi commit h vo master branch me show nhi honge jab aap “git status” command chalaoge tab

* git merge –squash FEATURE\_BRANCH\_NAME
* git merge – squash deepak

**STASH (छिपाना)**

git stash **temporarily shelves (or stashes) changes you've made to your working copy so you can work on something else, and then come back and re-apply them later on**.

* git stash (instead of commit command)
* git stash list (to show all stashed data)
* git show stash (to show that what work you have done)
* git stash apply (to remove stash from file)

IQ – Difference between squash and stash?

IQ – Difference between clone and pull?